Design of Localization System Based on Particle Filter Algorithm for Mobile Soccer Robot Using Encoders, Compass, and Omnidirectional Vision Sensor

Ahmad Wahrudin, Augie Widyotriatmo, and Endra Joelianto

Abstract—Self-localization is the basis to navigate robot or vehicle in dynamic environment such as for motion planning and obstacles avoidance. Self-localization can be divided into two categories: Local Localization-System (LLS) and Global Localization-System (GLS). Local Localization-System uses inertial sensors such as encoders which leads inevitably to the unbounded accumulation of errors. Whereas Global Localization-System utilizes information based on absolute sensors so that it has a long sample time. In the Middle Size Soccer Robots, the sensors unbounded accumulation of errors. Whereas Global Localization-System utilizes information based on absolute sensors so that it has a long sample time. In the Middle Size Soccer Robots, the sensors must be mounted on the robot so that it is difficult to obtain a global position directly. The particle filter algorithm is designed as a technique for combining both inertial and absolute sensor data to overcome the problems of Local Localization-System and Global Localization-System on mobile soccer robot. In this paper, three encoders are used to provide odometry motion model, an omni directional vision sensor is used to give weight to the particles, and ambiguity problems is overcome by using an electronic compass. The result of this test show that localization by using Particle Filter Algorithm gives better performance than Local Localization-System and can overcome the Global Localization-Problems.

Index Terms—GLS, LLS, Particle Filter, Self-Localization.

I. INTRODUCTION

MIDDLE Size League (MSL) is one of the branches in RoboCup which competes teams of five fully autonomous wheeled robots to play soccer using FIFA’s sized soccer ball. All the robots must truly represent human players, so they must be able to perceive the environment through sensors that must be installed on-board. The research is focused on autonomous multi-robot control, mechatronic, multi-agent cooperation, robot perception and navigation [1][2].

Manuscript received September 15, 2019.

This work was supported in part by Engineering Physic Department, Instrumentation and Control Research Group, Institut Teknologi Bandung, the Autonomus Vehicle Research Group, and the Dagozilla Mobile Soccer Robotics Team.

A. Wahrudin was with the Department Engineering Physics, Faculty of Industrial Technology, Institut Teknologi Bandung, Bandung (e-mail: ahmadwahrudin@gmail.com).

A. Widyotriatmo is with the Instrumentation and Control Research Group, Institut Teknologi Bandung, Bandung. (e-mail: augie@tf.itb.ac.id).

E. Joelianto is with the Instrumentation and Control Research Group, Institut Teknologi Bandung, Bandung, Indonesia (e-mail: ejoel@tf.itb.ac.id).

Self-localization is one of the most important issues in autonomous mobile robots, especially for the robots in the MSL competition that has high dynamic environment [3]. Self-localization can be divided into two categories: Local Localization-System and Global Localization-System. Local Localization-System uses inertial sensors such as encoders which leads inevitably to the unbounded accumulation of errors [4] [5]. Whereas Global Localization-System utilizes information based on absolute sensors such as Global Positioning System (GPS) and active beacons so that it has more accurate measurement results compared to the Local Localization-System method but requires a longer computational time [2].

This paper presents the design and implementation of particle filter algorithm include how to characterize the noise source, that is critical to obtain better performance of MSL Soccer Robot self-localization system. This article shows encoder modification to reduce slippage error and improve the sensor reading accuracy. Information generated from encoders is used as input to the motion model while compass and omnidirectional vision sensor is used as input to the measurement model in the particle filter algorithm.

II. MODIFIED ODOMETRY SENSOR

Odometry is the most commonly used as Local Localization-system. Odometry is obtained by calculating the incremental rotation of the wheel connected to the encoder according to its kinematics configuration. Because it only uses wheel rotation, there will always be an increase in reading errors continuously. This error is divided into 2 categories, systemic and non-systemic errors. Systematic errors are errors caused by imperfections in robot mechanics, such as differences in wheel diameter and unequal wheel distance to the center of the robot's geometry. Non-systematic errors are caused by wheel-floor interactions such as slips, bumps, and cracks [6].

In Robocup MSL, the main cause of non-systematic errors is the slip that occurs when changes in speed and direction of the robot motion [7]. A modification has been made to reduce this error by separating the encoder from the main wheel, then making it flexible by using a spring to ensure that the encoder wheel will always contact the ground as shown in Figure 1. This modification significantly reduce the accumulated errors of the
system.

**III. ROBOT KINEMATICS**

The basic modelling for our robot is shown in Fig. 2 which shows our robot model for three encoder wheels configuration with following notation [8] [6]:

- \(x, y, \theta\) : relative position of the robot in meter \((x, y)\) and angle in radian that defines the robot’s heading \((\theta)\) according to the field coordinate;
- \(L\) : Distance between wheels and center of robot’s geometry in meter;
- \(v_1, v_2, v_3\) : Encoder wheels linear velocity in m/s;
- \(\omega_1, \omega_2, \omega_3\) : Encoder wheels angular velocity in rad/s;
- \(v, \omega\) : Robot linear velocity in m/s;

Therefore, the linear velocity vector of the encoder wheel can be represented as a matrix function of the robot’s linear and angular velocity as shown (1)

\[
\begin{bmatrix}
v_1(t) \\
v_2(t) \\
v_3(t)
\end{bmatrix} = \frac{1}{r} \begin{bmatrix}
\frac{\sqrt{3}}{2} & -\frac{1}{2} & L \\
0 & 1 & L \\
-\frac{\sqrt{3}}{2} & -\frac{1}{2} & L
\end{bmatrix}
\begin{bmatrix}
v(t) \\
v(t) \\
\omega(t)
\end{bmatrix}
\]

Linear and angular velocities of the robot motion can be written as shown in (2) by calculating inverse matrix in (1).

By integrating (2), we can calculate current position and heading of the robot as shown in (3).

\[
x = \int \frac{1}{r} \left( \frac{\sqrt{3}}{2} v_1(t) - \frac{1}{2} v_2(t) - \frac{1}{2} v_3(t) \right) dt
\]

\[
y = \int \frac{1}{r} \left( \frac{\sqrt{3}}{2} v_2(t) + \frac{2}{3} v_3(t) - \frac{1}{3} v_1(t) \right) dt
\]

\[
\theta = \int \frac{1}{3L} \left[ v_1(t) + \frac{1}{3L} v_2(t) + \frac{1}{3L} v_3(t) \right] dt
\]

By transforming the relative positions from (2), we can estimate the global position of the robot according to equation 4.

\[
\begin{bmatrix}
X_{\text{new}} \\
Y_{\text{new}}
\end{bmatrix} = \begin{bmatrix}
\cos(\theta) & -\sin(\theta) \\
\sin(\theta) & \cos(\theta)
\end{bmatrix}
\begin{bmatrix}
X_{\text{prev}} \\
Y_{\text{prev}}
\end{bmatrix}
\]

**IV. PARTICLE FILTER**

Particle filter, also known as a Sequential Monte Carlo (SMC) method is an implementation of Bayes Filter which uses a Monte Carlo approach to represent the probability of a stochastic system at the present time in a Markov process. Each particle represents one of the hypotheses of the system state parameter. Each particle undergoes evolution and weighting based on a motion model and its measurement model with a certain error distribution [9].

The input of the particle filter algorithm is the set of particle \(X_{t-1} = \{X_{t-1}^{[1]}, X_{t-1}^{[2]}, \ldots, X_{t-1}^{[M]}\}\), the latest input control \(u_t\), and the latest measurement \(z_t\) [9]. The particle filter algorithm is shown in Fig. 3.

**Algorithm Particle filter**

\[
\text{Algorithm Particle filter}(X_{t-1}, u_t, z_t): \quad \hat{x}_t = x_0 = 0 \\
\text{for } m = 1 \text{ to } M \text{ do} \quad \text{sample } x_{t|m}^{[m]} \sim p(x_t | u_t, x_{t-1}^{[m]}) \\
\text{weight } w_{t|m}^{[m]} = p(z_t | x_{t|m}^{[m]}) \\
\text{endfor} \\
\text{for } m = 1 \text{ to } M \text{ do} \quad \hat{x}_t = \hat{x}_t + \left( x_{t|m}^{[m]} - \hat{x}_t \right) \text{ with probability } w_{t|m}^{[m]} \text{ add } x_{t|m}^{[m]} \text{ to } x_t \\
\text{endfor} \\
\text{return } x_t
\]

Fig. 3. Particle Filter Algorithm

The particle filter algorithm consists of several main process:

- Predicting process: predict new distributions for particles according to the robot motion model \(P(x_t | u_t, x_{t-1})\);
• Updating process: update the particle weights $w_i^{[m]}$ using information from sensor model $P(z_t | x_t)$ and then normalize the results: the localization result can be obtained by calculating the weighted mean over all particles.

• Resampling process: acquire a new set of $\chi$ according to particle weights: the probability for each particles $\chi_i^{[n]}$ to be resampled proportional to its weight $w_i^{[n]}$.

A. Odometry Motion Model

The odometry motion model algorithm is used as a source of motion model in prediction step. The input of this algorithm is previous state value $\chi_{i-1}$, and the input of the movement $u_t = (\delta_{\text{trans}}, \delta_{\text{rot1}}, \delta_{\text{rot2}})$ [9]. Where $\delta_{\text{trans}}$ is the lateral translation of the robot movement while $\delta_{\text{rot1}}$ and $\delta_{\text{rot2}}$ are the rotational movements taken by the robot shown by Fig. 4.

![Fig. 4. Odometry Model Illustration in Robot Linear Motion](image)

In the sample odometry motion model algorithm for omnidirectional wheel that has been modified from common odometry motion model [9] as shown in Fig. 5 there exist 5 parameters that represent noise parameter in the robot motion model. These parameter are listed below:

- $\alpha_1$: Specifies the expected noise in odometry's rotation estimate from the rotational component of the robot's motion;
- $\alpha_2$: Specifies the expected noise in odometry's rotation estimate from translational component of the robot's motion;
- $\alpha_3$: Specifies the expected noise in odometry's translation estimate from the translational component of the robot's motion;
- $\alpha_4$: Specifies the expected noise in odometry's translation estimate from the rotational component of the robot's motion;
- $\alpha_5$: Translation-related noise parameter that caused by omnidirectional robot characteristic.

![Fig. 5. Odometry Motion Model Algorithm](image)

B. Measurement Model

The measurement model $P(z_t | x_t)$ in the particle filter algorithm is obtained by processing the information from the omnidirectional vision sensor that captures $360^\circ$ image as shown in Fig. 6. The value of this measurement model is used as a weighting of each particle.

![Fig. 6. Panoramic Image Captured by Omni-Vision](image)

In the RoboCup MSL only the white lines of the field can be utilized as a measurement information source for localization process. So the radial scan lines method is applied to the image that has been calibrated for detecting the lines as shown in Fig. 7.

If $n$ line points are detected, the relative coordinates of the detected line points to the robot for each particle can be defined as $f_i = (o_{i1}, o_{i2})$, with $i = 1, 2, ..., n$. The term $P(f_i | \chi)$ is the probability of detecting $f_i$ when the robot is at $\chi_t = (x_t, y_t, \theta_t)$.

The position for each $f_i$ point in global coordinate can be determined by performing a geometry transformation according to (5).

![Fig. 7. Calibrated Image (Left) and Radial Scan Lines (Right)](image)

$$o_{i1} = \begin{pmatrix} x_i \\ y_i \end{pmatrix} + \begin{pmatrix} \cos \theta_i & -\sin \theta_i \\ \sin \theta_i & \cos \theta_i \end{pmatrix} o_{i1}^{[1]}$$

$$P(f_i | \chi) = \exp\left(-\frac{d(o_i) d(o_i)}{2\sigma^2}\right)$$

Where $\sigma$ is a constant. Therefore $P(f_i | \chi)$ can be calculated by the deviation between the distribution of $o_i$ and the actual position of these points on the field, and $P(f_i | \chi)$ decreases as the deviation increases to the closest line named as $d(o_i)$. So the deviation only depends on $o_i$, and it can be precalculated and stored in two-dimentional look-up table. Fig. 8 shows $d(o_i)$ distribution on the field. From Fig. 8, the brightness...
represents the deviation scale, and the higher brightness depicts the smaller deviation, so we can obviously investigate that how the deviation change with varying \( \sigma \) on the field.

![Image](image1.png)

**Fig. 8.** The Distribution of \( d(\sigma) \) on The Field

Because \( f_1, ..., f_n \) are detected independently, the sensor measurement model of the system can be represented as (7).

\[
P(z | x) = P(o | x) = P(f_1 | x) ... P(f_n | x)
\]

(7)

C. Resampling

The distribution of particles tends to degenerate where there particles with low weight due to dispersion in the prediction process. The selection / resampling stage is the key stage in the particle filter algorithm. The particle selection process needs to be carried out to keep the particle distribution in the correct posteriori distribution area.

Low Variance Sampler algorithm also called the systematic method is used for the resampling process. The algorithm is shown in Fig. 9. The probability for each particle \( x_i^m \) to be resampled is proportional to its weight \( w_i^{[m]} \). This algorithm converts the set of prior particles \( x_{i-1} \) into a set of posterior particles \( x_i \) and then rearranges the weight of the \( m \)-th particle to the same size \( w_i^{[m]} = \frac{1}{M} \), with \( M \) is the number of particles.

The low-variance sampler is relatively easier to implement and has a computational complexity \( O(M) \) which is faster than the independent selection method with complexity \( O(M \log M) \) [9]. After the selection process is finished, the program will run recursively until the program is stopped.

![Algorithm](algorithm.png)

**Fig. 9.** The Low Variance Sampler Algorithm

V. EXPERIMENTAL RESULTS

A. Sensor Characteristics

Compass

We use CMPS12 as the absolute orientation sensor. By giving an angle of 360° into the compass for 100 times testing, we can obtain the sensor characteristics as shown in the Fig. 10.

![Compass Measurement Errors Distribution](chart.png)

**Fig. 10.** Compass Measurement Errors Distribution

Normality test is carried out by using the Kolmogorov – Smirnov method at a significance level of 5%. From this test we can concluded that empirical data came from normally distributed populations as shown in Table I.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Standar Deviation (( \sigma ))</td>
<td>2.31405°</td>
</tr>
<tr>
<td>Measurement Model</td>
<td>( \theta_{cmps} = \theta_{cmps} + \sigma.N(0,1) )</td>
</tr>
</tbody>
</table>

Encoders

To obtain the encoders sensor characteristic when used as an odometry, the robot is moved in the translational direction along one meter, two meter, and three meter for 100 times testing. The measurements is recorded and plotted as shown in Fig. 11. The measurement errors is calculated and sensor characteristic value is obtained as indicated by Table II.

![Encoders Measurement Errors Distribution](chart2.png)

**Fig. 10.** Compass Measurement Errors Distribution

**TABLE I.**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>1 meter</th>
<th>2 meter</th>
<th>3 meter</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mean</td>
<td>0.02</td>
<td>0.06</td>
<td>0.05</td>
</tr>
<tr>
<td>Sttdev (Y)</td>
<td>0.012104</td>
<td>0.091311</td>
<td>0.173108</td>
</tr>
</tbody>
</table>

**TABLE II.**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>1 meter</th>
<th>2 meter</th>
<th>3 meter</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mean</td>
<td>0.02</td>
<td>0.06</td>
<td>0.05</td>
</tr>
<tr>
<td>Sttdev (Y)</td>
<td>0.012104</td>
<td>0.091311</td>
<td>0.173108</td>
</tr>
</tbody>
</table>
**B. Simulation Results**

**Odometry Motion Model**

The simulation of odometry motion model is aimed to obtain the best noise parameter value in odometry motion model algorithm. By using 100 set of particle test that represent 100 times of testing and varying the values of the noise parameters from 0.00 until 0.1 we can obtain a plot of data as shown in Fig. 12.

\[
F_{err} = |y_1 - Y_1| + |y_2 - Y_2| + |y_3 - Y_3|
\]  

where \(y_1, y_2, \) and \(y_3\) are measurement standard deviation from simulation models, and \(Y_1, Y_2, \) and \(Y_3\) are measurement standard deviation from the sensor realtime testing that we obtain before. From the calculation, the optimal noise parameter value is described as shown in Table III.

<table>
<thead>
<tr>
<th>Param</th>
<th>(\alpha_1)</th>
<th>(\alpha_2)</th>
<th>(\alpha_3)</th>
<th>(\alpha_4)</th>
<th>(\alpha_5)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Value</td>
<td>0.0026</td>
<td>0.0026</td>
<td>0.0026</td>
<td>0.0026</td>
<td>0.0026</td>
</tr>
</tbody>
</table>

**Particle Filter Simulation**

Particle filter simulation is carried out to determine the optimum particle filter parameters before being tested on the robot. Particle filter algorithm is performed on the S-shaped path as shown with a blue line in Fig. 13 (a) to simulate changes in velocity and direction. The result of Particle filter simulation is shown in Fig. 13(b). Red line defines the estimate position using filter particle, yellow line defines the estimate position using odometry and yellow dot points describe the detected line points with noise.

The performance of particle filters is determined by the correctness of the estimated results and the computational time required. Fig. 14(a) shows the plot of the particle filter estimation errors in the simulation varying by number of particles. The RMS (Root Mean Square) error shows how fast the filter estimation reaches convergence to the actual position. Fig. 14(b) shows the computational time required corresponding to the number of particles.

\[
err_{rms} = 933.11e^{0.01M}
\]

\[
t = 0.0001M - 0.0013
\]

To obtain the optimal value of the particle number, the optimization technique is carried out on the objective function describe in (11).

\[
F_M = |err_{rms}| + |vt|
\]

Where \(M\) is the particles number and \(v\) is the velocity given to the robot in the simulation. From the experiment, the optimal number of particles that can be used in the algorithm is \(M=645\) particles.

From Fig. 15, we can conclude that the computational time of the Particle Filter is dominated by weighting step. As we can see in (7), the weighting step depends on the number of \(n\) line points that being used in the algorithm. Graph of Particle Filter performance to the change of the number of \(n\) line points is shown in Fig. 16.
From the experiment, the optimal number of line points that can be used in the algorithm is $n=60$ line points.

From the simulation, by using parameter $M=645$ particles and $n=60$ points, a comparison of localization error simulation using odometry and filter particle is shown by Table IV.

**TABLE IV. ESTIMATION ERROR OF ROBOT LOCALIZATION**

<table>
<thead>
<tr>
<th>Method</th>
<th>Test Type</th>
<th>Position Error (cm)</th>
<th>Orientation Error</th>
</tr>
</thead>
<tbody>
<tr>
<td>LLS /Odometry</td>
<td>RMS</td>
<td>64.488</td>
<td>0.119°</td>
</tr>
<tr>
<td>Particle</td>
<td>Final</td>
<td>110.477</td>
<td>11.923°</td>
</tr>
<tr>
<td>Filter</td>
<td>Final</td>
<td>3.932</td>
<td>0.014°</td>
</tr>
</tbody>
</table>

**Realtime Implementation**

Particle Filter algorithm has been implemented in the robot by using the optimal parameters from simulation. The self-localization using Particle filter starts with initializes the particles position over the field uniformly, and uses data from the compass as the orientation initialization to eliminate ambiguity problem.

The experiment is carried out by moving the robot in the field on free paths and with accelerated motion upto maximum velocity 1.5m/s. The robot is driven through the control of the GUI system from basestation.

Fig. 17(a) shows the initial position and Fig. 17(b) shows the final position. Where the blue line shows historical position estimation data using odometry, while the red line is a historical position estimation data using Particle Filter, and green dots are the particles.
The self-localization system using Particle filter can be implemented on MSL mobile soccer robot by using encoders, compass, and omni-directional vision sensor. The algorithm has optimal parameter \( M = 645 \) particles and \( n = 60 \) points. Localization system using Particle Filter has better performance than using LLS (odometry) method with estimated simulation error value of Particle filter of 3 cm while on the LLS of 110 cm.

VI. CONCLUSION

The self-localization system using Particle filter can be implemented on MSL mobile soccer robot by using encoders, compass, and omni-directional vision sensor. The algorithm has optimal parameter \( M = 645 \) particles and \( n = 60 \) points. Localization system using Particle Filter has better performance than using LLS (odometry) method with estimated simulation error value of Particle filter of 3 cm while on the LLS of 110 cm.

VII. ACKNOWLEDGMENT

The authors would like to thank the Institut Teknologi Bandung Engineering Physics Department, the Instrumentation and Control Research Group, the Autonomous Vehicle Research Group, and the Dagozilla Mobile Soccer Robotics Team for the help and support so this paper could be finished.

REFERENCES


Ahmad Wahrudin received his B.Eng. in engineering physics from the Institut Teknologi Bandung, Indonesia in 2019.
Since 2016, he was a research fellow with the Autonomous Vehicle Research Group, Institut Teknologi Bandung, and member of ITB Dagozilla Mobile Soccer Robotics Team. His research interests include robotics, autonomous vehicles, automation system, and data analytic.

Augie Widyotriatmo (M’08) received his B.Eng. in engineering physics and M. Eng in instrumentation and control from the Institut Teknologi Bandung, Indonesia in 2002 and 2006, respectively, and Ph.D. degree in mechanical engineering from Pusan National University, South Korea in 2011.
He joined the Faculty of Industrial Technology as a faculty member for engineering physics program in the Institut Teknologi Bandung Indonesia in 2010. He received honors of contribution in innovations from Institut Teknologi Bandung in 2016 for his invention on robotic wheelchair and in 2018 on cockroach robots. He also received the outstanding paper award in the 18th International Conference on Control, Automation and Systems (ICCAS 2018), South Korea, best paper awards in the 4th International Conference on Industrial Internet of Things 2018 (ICIOT 2018), Indonesia, and in the 3rd International Conference on Robotics, Biomimetics, and Intelligent Computational Systems 2018 (ROBIONETICS 2018), Indonesia.
He was a visiting scholar in the University of Melbourne, Australia, in 2016. Dr. Widyotriatmo was the chair of the IEEE Indonesia Control Systems and Robotics and Automation Joint Chapter Societies for the period of 2017-2018. He is an associate editor in the International Journal of Control, Automation, and Systems (IJCAS). His research interests include robotics, autonomous vehicles, nonlinear control, energy optimization, biomedical instrumentation, automation system, and power system control.

Endra Joelianto (M’07–SM’19) received his B.Eng. in engineering physics from the Institut Teknologi Bandung, Indonesia in 1990, and his Ph.D. degree in Engineering from The Australian National University (ANU), Australia in 2002.
He has been with the Engineering Physics Study Program, Institut Teknologi Bandung, Indonesia as faculty member, since 1999. He has been a senior research fellow at the Centre for Unmanned System Studies (CENTRUMS), Bandung Institute of Technology, Bandung, Indonesia since 2007. He was a visiting scholar in the Telecommunication and Information Technology Institute (TTIT), University of Wollongong, NSW, Australia in February 2002. He was a Visiting Scholar in the Smart Robot Center, Department of Aerospace and Information Engineering, Konkuk University, Seoul, Korea in October 2010.
He has edited one book on intelligent unmanned systems (Springer, 2009), published one book on Linear Quadratic Control: A State Space Approach (ITB Press, 2017) and published more than 125 research papers. Dr. Joelianto is a Senior Member IEEE. He was the chair of the IEEE Indonesia Control.
He is the Chairman of Society of Automation, Control and Instrumentation, Indonesia. He is an Editor of the International Journal of Artificial Intelligence (IJAI) and the International Journal of Intelligent Unmanned Systems (IJIUS). He was the Guest Editor at the International Journal of Artificial Intelligence (IJAI), the International Journal of Imaging and Robotics (IJIR) and the International Journal of Applied Mathematics and Statistics (IJAMAS). His research interest includes hybrid control systems, discrete event systems, computational intelligence, robust control, unmanned systems and intelligent automation.